There also may be a remote procedure call (RPC) subsystem on the computer system that facilitates remote processing requests between the input device, the computer system, and remote computers connected to the network. For example, the RPC subsystem enables software applications executing on the computer system to display keyboard overlays on the keys of the input device. Remote method invocation (RMI) developed by Sun Microsystems, Inc. is one such RPC subsystem capable of providing these features. Those skilled in the art, however, will appreciate that other RPC subsystems, such as DCOM/COM from Microsoft, Inc., may be used.

SYSTEM CONFIGURATION

FIG. **1** is block diagram of an exemplary system **100** with which methods and systems consistent with the present invention may be implemented. System **100** includes a computer **101** and a keyboard device **110**. Computer **101** includes a memory **102**, a CPU **104**, a network interface **106** to connect to a network **108**, and a bus **107** that provides connectivity and communication among these components. Bus **107** uses a bus protocol such as ISA, PCI, or SCSI. Network **108** may be a Local Area Network (LAN), a Wide Area Network (WAN), or the Internet.

Memory **102** includes an application **112** and a runtime system **116**. A user may need a special keyboard layout when executing application **112**. For example, a game application may use a special set of keys on a keyboard to interact with the game. Internationalized applications that operate in different languages may also need a special set of keys on a keyboard corresponding to the alphabet of a particular language.

Runtime system **116** provides an execution environment that enables computer system **101** to process application **112**. In one embodiment, runtime system **116** includes a virtual machine **120**, such as the Java™ Virtual Machine, and an RPC subsystem **118** such as RMI. Application **112** may utilize an Application Programming Interface (API) to access runtime system **116** and the various subsystems in a platform-independent manner. The Java™ Virtual Machine, RMI, and API are provided as part of the Java™ Development Kit from Sun Microsystems, Inc. of Mountain View, Calif.

Virtual machine **120** facilitates platform independence. Virtual machine **120** is an abstract computing machine that receives instructions from programs in the form of bytecodes. These bytecodes are interpreted and dynamically converted into a form for execution, such as object code, on a processor such as CPU **104**. Virtual machine **120** can be a process in memory **102** simulating execution of instructions of a virtual machine or it can be an integrated circuit processor designed to be compatible with the architecture of virtual machine **120**.

RPC **118** facilitates remote method invocation. Remote method invocation allows a process executing on one device to invoke a method or procedure associated with a process executing on another device. Typically a network connected between the two computers facilitates communication necessary to perform the remote method invocation.

Keyboard input device **110** includes a processor complex **111** and selectable keyboard display elements **132**. Processor complex **111** includes a memory **126**, a display processor **129**, a CPU **127**, and a non-volatile random access memory (NVAM) **128**. Each component in processor complex **111** may be a collection of discrete processing subsystems or may be a processor on an integrated circuit (IC) capable of processing keystrokes and driving selectable keyboard display elements **132**.

Each keyboard display element **132** displays a symbol. In one implementation, one selectable keyboard display element **132** can be an electro-mechanical device actuated when the user depresses and releases the device. A display device on each selectable keyboard display element **132** indicates which symbol is generated.

A smartcard reader **134** may be connected to a bus, such as a serial bus, on keyboard **110**. This smartcard reader interfaces with a smartcard device **135**. Smartcard device **135** can hold a useres preferences associated with configuring computer system **101** and may also include a keyboard applet or a user's preferred keyboard layout. For example, smartcard device **135** can define the language that selectable keyboard display elements **132** should display and the keys for displaying special functions for file management operations, macro invocations, and other often used functions in applications such as wordprocessors.

Memory **126** includes a keyboard applet **114**, a keyboard layout **115**, a runtime system **125**, such as the Java™ runtime environment, a virtual machine **122**, such as the Java™ virtual machine, and an RPC **123** subsystem. Subsystems in memory **126** operate in a similar manner to like named subsystems discussed previously. RPC **123** and RPC **118** enable application **112** to invoke methods associated with keyboard applet **114** executing on keyboard **110**. Applets, such as keyboard applet **114**, are modular software components that perform a subset of functions in a software application. The applet can be written in a procedural programming language such as C or an object-oriented language such as the Java™ programming language. Typically, virtual machine **122** is used to process methods associated with keyboard applet **114**. For example, actuating a key on keyboard **110** causes applet **114** to send a keyboard symbol in the form of a signal back to application **112** for further processing. This enables application **112** to distribute execution of instructions on CPU **104** as well as CPU **127**.

A keyboard layout **115** provides the data to indicate the symbols generated when actuating a key on keyboard **110**. Technically, a user actuates a key on a keyboard by depressing a key, releasing a key, or depressing and releasing a key or combination of keys on the keyboard. In one implementation consistent with the present invention, keyboard layout **115** may include a look-up table that maps certain keys to certain functions in an application. By changing the keyboard layout **115**, a keyboard **110** has the capability of generating different symbols on the keycaps.

Keyboard applet **114** can be used to process keyboard layout **115** in several ways. In one implementation consistent with the present invention, each keyboard applet contains a different keyboard layout. To change a keyboard layout, computer system **100** downloads a different keyboard applet containing the new keyboard layout from either host computer **101**, network **108**, or smart card **134**. The keyboard applet containing the keyboard layout such as keyboard layout **115** displays the appropriate characters on selectable keyboard display elements **132**. In an alternative implementation consistent with the present invention, keyboard applet **114** and keyboard layouts are stored separately on, for example, different parts of network **108**. In this implementation, one keyboard applet can be used to process many different keyboard applets downloaded over network **108**.

In an object-oriented programming environment, a class loader mechanism, such as the class loader used for the Java™ programming language, may be used to locate and download the appropriate keyboard applet, keyboard layout, and related object classes automatically. Additional infor-